

EV355227682

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

**Methods And Apparatuses For Providing Short Digital  
Signatures Using Curve-Based Cryptography**

Inventor(s):

**Ramarathnam Venkatesan**

**Dan Boneh**

ATTORNEY'S DOCKET NO. MS1-1043US

## **TECHNICAL FIELD**

This invention relates to cryptography, and more particularly to cryptography systems, apparatuses and related methods that provide and/or use short digital signatures based on curve-based cryptography techniques.

## **BACKGROUND**

As computers have become increasingly commonplace in homes and businesses throughout the world, and such computers have become increasingly interconnected via networks (such as the Internet), security and authentication concerns have become increasingly important. One manner in which these concerns have been addressed is the use of a cryptographic technique involving a key-based cipher. Using a key-based cipher, sequences of intelligible data (typically referred to as plaintext) that collectively form a message are mathematically transformed, through an enciphering process, into seemingly unintelligible data (typically referred to as cipher text). The enciphering can be reversed, allowing recipients of the cipher text with the appropriate key to transform the cipher text back to plaintext, while making it very difficult, if not nearly impossible, for those without the appropriate key from recovering the plaintext.

Public-key cryptographic techniques are one type of key-based cipher. In public-key cryptography, each communicating party has a public/private key pair. The public key of each pair is made publicly available (or at least available to others who are intended to send encrypted communications), but the private key is kept secret. In order to communicate a plaintext message using encryption to a receiving party, an originating party encrypts the plaintext message into a cipher

1 text message using the public key of the receiving party and communicates the  
2 cipher text message to the receiving party. Upon receipt of the cipher text  
3 message, the receiving party decrypts the message using its secret private key, and  
4 thereby recovers the original plaintext message.

5 The RSA (Rivest-Shamir-Adleman) method is one well-known example of  
6 public/private key cryptology. To implement RSA, one generates two large prime  
7 numbers  $p$  and  $q$  and multiplies them together to get a large composite number  $N$ ,  
8 which is made public. If the primes are properly chosen and large enough, it will  
9 be practically impossible (i.e., computationally infeasible) for someone who does  
10 not know  $p$  and  $q$  to determine them from just knowing  $N$ . However, in order to  
11 be secure, the size of  $N$  typically needs to be more than 1,000 bits. In some  
12 situations, though, such a large size makes the numbers too long to be practically  
13 useful.

14 One such situation is found in authentication, which can be required  
15 anywhere a party or a machine must prove that it is authorized to access or use a  
16 product or service. An example of such a situation is in a product ID system for a  
17 software program(s), where a user must enter a product ID sequence stamped on  
18 the outside of the properly licensed software package as proof that the software  
19 has been properly paid for. If the product ID sequence is too long, then it will be  
20 cumbersome and user unfriendly.

21 Additionally, not only do software manufacturers lose revenue from  
22 unauthorized copies of their products, but software manufacturers also frequently  
23 provide customer support, of one form or another, for their products. In an effort  
24 to limit such support to their licensees, customer support staffs often require a user  
25 to first provide the product ID associated with his or her copy of the product for

1 which support is sought as a condition for receiving support. Many current  
2 methods of generating product IDs, however, have been easily discerned by  
3 unauthorized users, allowing product IDs to be generated by unauthorized users.

4         Given the apparent ease with which unauthorized users can obtain valid  
5 indicia, software manufacturers are experiencing considerable difficulty in  
6 discriminating between licensees and such unauthorized users in order to provide  
7 support to the former while denying it to the latter. As a result, manufacturers  
8 often unwittingly provide support to unauthorized users, thus incurring additional  
9 and unnecessary support costs. If the number of unauthorized users of a software  
10 product is sufficiently large, then these excess costs associated with that product  
11 can be quite significant.

12         New curve-based cryptography techniques have recently been employed to  
13 allow software manufacturers to appreciably reduce the incidence of unauthorized  
14 copying of software products. For example, product IDs have been generated  
15 using genus one elliptical curve-based cryptography techniques. It would be  
16 beneficial to be able to utilize higher order genus curves, e.g., hyperelliptic curves  
17 with genus greater than one as doing so will likely further improve security.  
18 Moreover, it would be beneficial for the resulting information (data) to have a size  
19 that is suitable for use as a short digital signature, product ID, and/or the like.

## 20 21 **SUMMARY**

22         In accordance with certain implementations of the present invention, a  
23 method is provided that includes identifying data that is to be signed, and  
24 establishing parameter data for use with signature generating logic that encrypts  
25 data based on a Jacobian of genus exceeding one. Here, parameter data causes the

1 signature generating logic to select at least one Gap Diffie-Hellman (GDH) group  
2 of elements relating to the curve. The method further includes determining private  
3 key data and corresponding public key data and signing the identified data with  
4 the private key data using the signature generating logic to create a corresponding  
5 digital signature.

6 An exemplary apparatus includes memory that is configured to store  
7 identifying data that is to be signed and signature generating logic that encrypts  
8 data based on a Jacobian of at least one curve according to the above method.

9 In accordance with certain other exemplary implementations of the present  
10 invention, a method includes receiving message data and a corresponding digital  
11 signature and public key data, and using parameter data configure signature  
12 verifying logic that performs cryptography operations based on a Jacobian of at  
13 least one curve, the parameter data causing the signature verifying logic to select  
14 at least one Gap Diffie-Hellman (GDH) group of elements relating to the curve.  
15 The method also includes using the signature verifying logic to determine if the  
16 digital signature is valid using the public key data and the message data.

17 In accordance with still other exemplary implementations, a method is  
18 provided that includes identifying data that is to be signed, and establishing  
19 parameter data for use with signature generating logic that encrypts data based on  
20 a Weil pairing on a Jacobian of at least one super-singular curve having a genus  
21 greater than one. The method also includes determining private key data and  
22 corresponding public key data using the signature generating logic, and signing the  
23 identified data with the private key data using the signature generating logic to  
24 create a corresponding digital signature.

1 In still further implementations, an apparatus having memory configured to  
2 store identifying data to be signed and signature generating logic that is configured  
3 using parameter data such that the signature generating logic encrypts data based  
4 on a Weil pairing on a Jacobian of at least one super-singular curve having a genus  
5 greater than one, and determines private key data and corresponding public key  
6 data and signs the identified data with the private key data using the signature  
7 generating logic to create a corresponding digital signature.

## 8 9 **BRIEF DESCRIPTION OF THE DRAWINGS**

10 The present invention is illustrated by way of example and not limitation in  
11 the figures of the accompanying drawings. The same numbers are used  
12 throughout the figures to reference like components and/or features.

13 Fig. 1 is a block diagram depicting an exemplary computing environment  
14 that is suitable for use with certain implementations of the present invention.

15 Fig. 2 is a block diagram depicting a cryptographic system in accordance  
16 with certain exemplary implementations of the present invention.

17 Fig. 3 is a flow diagram illustrating an exemplary cryptography process in  
18 accordance with certain implementations of the present invention.

## 19 20 **DETAILED DESCRIPTION**

### 21 **Introduction:**

22 In accordance with certain aspects of the present invention curve-based  
23 cryptography techniques are provided for use in systems, apparatuses and  
24 methods.

1 Many of these techniques are based on the Computational Diffie-Hellman  
2 assumption on certain high genus order (e.g., genus greater than one) hyper elliptic  
3 curve groups. The resulting encryption is believed to be at least as strong as that  
4 produced by a conventional Digital Signature Algorithm (DSA) for a similar level  
5 of security.

6 Short digital signatures are often used in environments where a user is  
7 asked to manually input a digital signature. For example, product registration  
8 systems often ask users to key in a digital signature provided on a CD label. More  
9 generally, short digital signatures are also useful in low bandwidth communication  
10 environments. For example, short digital signatures may be used when printing a  
11 digital signature on a postage stamp.

12 Currently, the two most frequently used digital signatures schemes, RSA  
13 and DSA, provide relatively long digital signatures (compared to the security they  
14 provide). For example, using a 1024-bit modulus, RSA digital signatures are 1024  
15 bits long. Similarly, using a 1024-bit modulus, standard DSA digital signatures are  
16 320 bits long. Elliptic curve variants of DSA, such as ECDSA, are also 320 bits  
17 long. For example see ANSI X9.62 and FIPS 186-2. Elliptic Curve Digital  
18 Signature Algorithm, 1998. A 320-bit digital signature may be too long to be  
19 keyed in by a user.

20 In accordance with certain exemplary implementations of the present  
21 invention, a digital signature scheme is provided that produces digital signatures  
22 having even shorter lengths, e.g., approximately 160 bits in certain instances, but  
23 which provides a similar level of security as longer 320-bit DSA digital signatures.  
24 Here, the digital signature scheme is secure against existential forgery under a  
25 chosen message attack (in the random oracle model) assuming the Computational

1 Diffie-Hellman (CDH) problem is hard on certain hyper elliptic curves over a  
2 finite field. Generating a digital signature, for example, can be as simple as  
3 multiplying on the hyper elliptic curve. Verifying the resulting digital signature  
4 can be accomplished using a bilinear pairing on the curve.

#### 5 6 Exemplary Operational Environment:

7 Turning to the drawings, wherein like reference numerals refer to like  
8 elements, the invention is illustrated as being implemented in a suitable computing  
9 environment. Although not required, the invention will be described in the general  
10 context of computer-executable instructions, such as program modules, being  
11 executed by a personal computer.

12 Generally, program modules include routines, programs, objects,  
13 components, data structures, etc. that perform particular tasks or implement  
14 particular abstract data types. Those skilled in the art will appreciate that the  
15 invention may be practiced with other computer system configurations, including  
16 hand-held devices, multi-processor systems, microprocessor based or  
17 programmable consumer electronics, network PCs, minicomputers, mainframe  
18 computers, portable communication devices, and the like.

19 The invention may also be practiced in distributed computing environments  
20 where tasks are performed by remote processing devices that are linked through a  
21 communications network. In a distributed computing environment, program  
22 modules may be located in both local and remote memory storage devices.

23 Fig.1 illustrates an example of a suitable computing environment 120 on  
24 which the subsequently described systems, apparatuses and methods may be  
25 implemented. Exemplary computing environment 120 is only one example of a

1 suitable computing environment and is not intended to suggest any limitation as to  
2 the scope of use or functionality of the improved methods and systems described  
3 herein. Neither should computing environment 120 be interpreted as having any  
4 dependency or requirement relating to any one or combination of components  
5 illustrated in computing environment 120.

6 The improved methods and systems herein are operational with numerous  
7 other general purpose or special purpose computing system environments or  
8 configurations. Examples of well known computing systems, environments,  
9 and/or configurations that may be suitable include, but are not limited to, personal  
10 computers, server computers, thin clients, thick clients, hand-held or laptop  
11 devices, multiprocessor systems, microprocessor-based systems, set top boxes,  
12 programmable consumer electronics, network PCs, minicomputers, mainframe  
13 computers, distributed computing environments that include any of the above  
14 systems or devices, and the like.

15 As shown in Fig. 1, computing environment 120 includes a general-purpose  
16 computing device in the form of a computer 130. The components of computer  
17 130 may include one or more processors or processing units 132, a system  
18 memory 134, and a bus 136 that couples various system components including  
19 system memory 134 to processor 132.

20 Bus 136 represents one or more of any of several types of bus structures,  
21 including a memory bus or memory controller, a peripheral bus, an accelerated  
22 graphics port, and a processor or local bus using any of a variety of bus  
23 architectures. By way of example, and not limitation, such architectures include  
24 Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA)  
25 bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA)

1 local bus, and Peripheral Component Interconnects (PCI) bus also known as  
2 Mezzanine bus.

3 Computer 130 typically includes a variety of computer readable media.  
4 Such media may be any available media that is accessible by computer 130, and it  
5 includes both volatile and non-volatile media, removable and non-removable  
6 media.

7 In Fig. 1, system memory 134 includes computer readable media in the  
8 form of volatile memory, such as random access memory (RAM) 140, and/or non-  
9 volatile memory, such as read only memory (ROM) 138. A basic input/output  
10 system (BIOS) 142, containing the basic routines that help to transfer information  
11 between elements within computer 130, such as during start-up, is stored in ROM  
12 138. RAM 140 typically contains data and/or program modules that are  
13 immediately accessible to and/or presently being operated on by processor 132.

14 Computer 130 may further include other removable/non-removable,  
15 volatile/non-volatile computer storage media. For example, Fig. 1 illustrates a  
16 hard disk drive 144 for reading from and writing to a non-removable, non-volatile  
17 magnetic media (not shown and typically called a "hard drive"), a magnetic disk  
18 drive 146 for reading from and writing to a removable, non-volatile magnetic disk  
19 148 (e.g., a "floppy disk"), and an optical disk drive 150 for reading from or  
20 writing to a removable, non-volatile optical disk 152 such as a CD-ROM/R/RW,  
21 DVD-ROM/R/RW/+R/RAM or other optical media. Hard disk drive 144,  
22 magnetic disk drive 146 and optical disk drive 150 are each connected to bus 136  
23 by one or more interfaces 154.

24 The drives and associated computer-readable media provide nonvolatile  
25 storage of computer readable instructions, data structures, program modules, and

1 other data for computer 130. Although the exemplary environment described  
2 herein employs a hard disk, a removable magnetic disk 148 and a removable  
3 optical disk 152, it should be appreciated by those skilled in the art that other types  
4 of computer readable media which can store data that is accessible by a computer,  
5 such as magnetic cassettes, flash memory cards, digital video disks, random access  
6 memories (RAMs), read only memories (ROM), and the like, may also be used in  
7 the exemplary operating environment.

8 A number of program modules may be stored on the hard disk, magnetic  
9 disk 148, optical disk 152, ROM 138, or RAM 140, including, e.g., an operating  
10 system 158, one or more application programs 160, other program modules 162,  
11 and program data 164.

12 The improved methods and systems described herein may be implemented  
13 within operating system 158, one or more application programs 160, other  
14 program modules 162, and/or program data 164.

15 A user may provide commands and information into computer 130 through  
16 input devices such as keyboard 166 and pointing device 168 (such as a "mouse").  
17 Other input devices (not shown) may include a microphone, joystick, game pad,  
18 satellite dish, serial port, scanner, camera, etc. These and other input devices are  
19 connected to the processing unit 132 through a user input interface 170 that is  
20 coupled to bus 136, but may be connected by other interface and bus structures,  
21 such as a parallel port, game port, or a universal serial bus (USB).

22 A monitor 172 or other type of display device is also connected to bus 136  
23 via an interface, such as a video adapter 174. In addition to monitor 172, personal  
24 computers typically include other peripheral output devices (not shown), such as  
25

1 speakers and printers, which may be connected through output peripheral interface  
2 175.

3 Computer 130 may operate in a networked environment using logical  
4 connections to one or more remote computers, such as a remote computer 182.  
5 Remote computer 182 may include many or all of the elements and features  
6 described herein relative to computer 130.

7 Logical connections shown in Fig. 1 are a local area network (LAN) 177  
8 and a general wide area network (WAN) 179. Such networking environments are  
9 commonplace in offices, enterprise-wide computer networks, intranets, and the  
10 Internet.

11 When used in a LAN networking environment, computer 130 is connected  
12 to LAN 177 via network interface or adapter 186. When used in a WAN  
13 networking environment, the computer typically includes a modem 178 or other  
14 means for establishing communications over WAN 179. Modem 178, which may  
15 be internal or external, may be connected to system bus 136 via the user input  
16 interface 170 or other appropriate mechanism.

17 Depicted in Fig. 1, is a specific implementation of a WAN via the Internet.  
18 Here, computer 130 employs modem 178 to establish communications with at  
19 least one remote computer 182 via the Internet 180.

20 In a networked environment, program modules depicted relative to  
21 computer 130, or portions thereof, may be stored in a remote memory storage  
22 device. Thus, e.g., as depicted in Fig. 1, remote application programs 189 may  
23 reside on a memory device of remote computer 182. It will be appreciated that the  
24 network connections shown and described are exemplary and other means of  
25 establishing a communications link between the computers may be used.

1  
2 Exemplary System and Apparatuses:

3       The description that follows assumes a basic understanding of cryptography  
4 by the reader. For a basic introduction of cryptography, the reader is directed to  
5 “Applied Cryptography: Protocols, Algorithms, and Source Code in C,” Second  
6 Edition, written by Bruce Schneier and published by John Wiley & Sons in 1996,  
7 and which is incorporated herein by reference in its entirety.

8       Attention is now directed to Fig. 2, which is a block diagram of a system  
9 200 that provides for short digital signature operations, in accordance with certain  
10 exemplary implementations of the present invention.

11       System 200 includes a first device 202 that is configured to generate a short  
12 digital signature that can then be provided to a second device 204 and verified.  
13 First device 202 includes curve-based cryptography signature generating logic  
14 206, which is configured according to parameter data 208. Once configured logic  
15 206 takes message data 210, for example, containing licensing information, etc.,  
16 and generates a corresponding digital signature 216. Digital signature 216 is  
17 generated based on the curve-based encrypting techniques provided herein, which  
18 include generating a secret or private key 212 and a corresponding public key 214.

19       Digital signature 216 is then provided, e.g., communicated, input, etc., to  
20 curve-based cryptography signature verifying logic 218 within second device 204.  
21 Here, logic 218 is also provided with message data 210, parameter data 208, and  
22 public key 214. Logic 218 then verifies digital signature 216 in accord with the  
23 verification schemes described herein. Thus, for example, in certain  
24 implementations logic 206 and 218 are configured to support GDH digital  
25 signature schemes, while in other implementations they are configured to support

1 a modified (co-gap) digital signature scheme. These schemes are described in  
2 greater detail below.

#### 3 4 Exemplary Signature Process:

5 Attention is now drawn to Fig. 3, which is a flow diagram depicting a short  
6 digital signature operation process 300, in accordance with certain exemplary  
7 implementations of the present invention. As with the block diagrams in Figs 1  
8 and 2, the flow diagram in Fig. 3 is configured to support/implement curve-based  
9 short digital signature processes for curves as described herein.

10 In act 302, curve-based cryptography signature generating logic is  
11 configured using parameter data. In act 304, a private key and a corresponding  
12 public key are generated using the curve-based cryptography signature generating  
13 logic. Then, in act 306, a digital signature for message data is generated using the  
14 private key.

15 The digital signature, message data and public key are then provided some  
16 manner(s) to curve-based cryptography signature verifying logic in act 308. For  
17 example, the message data and public key may be provided on computer-readable  
18 media, downloaded, etc., and the digital signature input by a user who is able to  
19 read the digital signature in the form of corresponding ASCII or other like encoded  
20 characters as printed on a package, sent via e-mail, etc., associated with the  
21 purchasing/licensing of a product. Hence, in certain implementations the digital  
22 signature may serve as a product ID.

23 In act 310, a determination is made as to whether the digital signature  
24 provided in act 308 is valid. This can be done with curve-based cryptography  
25 signature verifying logic. The curve-based cryptography signature verifying logic

1 can be configured with the parameter data, for example, and utilize the message  
2 data and public key as also provided in act 308.

#### 3 4 Exemplary Use Of High Genus Curves:

5 In accordance with certain aspects of the present invention curve-based  
6 cryptography techniques are provided for use in the exemplary systems,  
7 apparatuses and methods as described above, and others like them.

#### 8 9 Defining Gap-Diffie-Hellman Groups:

10 In accordance with certain aspects of the present invention, short digital  
11 signature schemes are provided that works in any Gap Diffie-Hellman (GDH)  
12 group (which is written multiplicatively when defined over the set of integers  
13 modulo a prime and written additively when the group is defined by the points on  
14 an elliptic curve or a Jacobian), as defined below, for example. These new  
15 constructions are based on giving new gap Diffie-Hellman groups.

16 Consider a (multiplicative) cyclic group  $G = \langle g \rangle$ , with  $p = |G|$  a prime.  
17 There three problems of interest on  $G$ , namely Group Action, Decision Diffie-  
18 Hellman and Computational Diffie-Hellman. We write  $g^a$  for the group element  
19 obtained by multiplying  $g$  by itself  $a$  times.

#### 20 Group Action:

21 Given  $u, v \in G$ , find  $uv$ .

#### 22 Decision Diffie-Hellman:

23 For  $a, b, c \in \mathbb{Z}_p^*$ , given  $g^a, g^b$ , and  $g^c$ , decide whether  $c = ab$ .

#### 24 Computational Diffie-Hellman (CDH):

25 For  $a, b \in \mathbb{Z}_p^*$ , given  $g^a$  and  $g^b$ , compute  $g^{ab}$ .

1 A Gap Diffie-Hellman (GDH) group can be defined in stages:

2 Let  $G$  be a  $\tau$ -decision group for Diffie-Hellman if the group action can be  
3 computed in one time unit, and Decision Diffie-Hellman can be computed in time  
4 at most  $\tau$ . This task is easy in a Gap DH group but the computational DH is  
5 considered infeasible.

6  
7 GDH Digital Signature Schemes:

8 An exemplary GDH digital signature scheme allows the creation of digital  
9 signatures on arbitrary messages  $m \in \{0, 1\}^*$ . Here, a digital signature  $\sigma$  is an  
10 element of  $G$ . The base group  $G$  and the generator  $g$  are system parameters (e.g.,  
11 included in parameter data 208 (Fig. 2)).

12 The digital signature scheme includes three basic algorithms, namely a key  
13 generation algorithm, a signing algorithm, and verifying algorithm. In certain  
14 implementations, the digital signature scheme makes use of a full-domain hash  
15 function  $h: \{0, 1\}^* \rightarrow G$ . In other implementations, for example as described in  
16 subsequent sections herein, the requirement on the full-domain hash may be  
17 weakened.

18 Key Generation:

19 Pick  $x \in \mathbb{Z}_p^*$ , and compute  $v \leftarrow g^x$ . Here, the public key is  $v$ ; the secret key is  
20  $x$ .

21 Signing:

22 Given a secret key  $x$ , and a message  $m \in \{0, 1\}^*$ , compute  $h \leftarrow h(m)$ , and  
23  $\sigma \leftarrow h^x$ . The digital signature is  $\sigma$ .

24 Verification:

25

1        Given a public key  $v$ , a message  $m$ , and a digital signature  $\sigma$ . Compute  
2         $h \leftarrow h(m)$ . Verify that  $(g, v, h, \sigma)$  is a valid Diffie-Hellman tuple.

3        Note that a GDH digital signature is a single element of  $G$ . Hence, to  
4        construct short digital signatures preferably the GDH group includes elements  
5        having short representations.

### 6        Extending the Signature Scheme To Use “Unreliable” Hashing:

8        The exemplary schemes presented above assume the existence of a hash  
9        function  $h$  that maps uniformly from arbitrary strings to elements of the GDH  
10       group. Such a function may not always be practical and/or immediately available.  
11       For example, hashing onto a subgroup of an elliptic curve over a finite field  
12       requires some care in order to maintain the proof of security.

13       More generally, it is possible that one only has an unreliable hash function  
14        $h': \{0, 1\}^* \rightarrow G \cup \{\perp\}$ . For a given message  $m \in \{0, 1\}^*$ , the hash function  $h'$   
15       outputs either an element of  $G$ , or  $\perp$  (the latter indicating a failure). For example,  
16       let  $h$  be an auxiliary hash function mapping messages in  $\{0, 1\}^*$  onto  $F_p$ . Then  
17        $h(m)$  outputs failure if  $h(m)$  is not an  $x$ -coordinate of any point in  $E/F_p$ . Otherwise  
18        $h'(m)$  outputs one of the points whose  $x$ -coordinate is  $h(m)$ . In the security  
19       analysis one may view  $h$  as a random oracle.

20       Let  $B \subseteq A$  be two finite sets with  $|B| = |G|$ . An “unreliable” hash function  
21        $h'$  is a composition of two functions:  $h'(m) = f(h(m))$ , where  $h: \{0, 1\}^* \rightarrow A$ . For  
22        $x \notin B$  we have  $f(x) = \perp$ . For  $x \in B$  the function  $f$  is one-to-one onto  $G$ . We say  
23       that  $h'$  is  $\eta$ -unreliable if  $|B|/|A| = \eta$ .

24       Note that for any  $m$ , an  $\eta$ -unreliable hash function  $h'$  satisfies  $h'(m) \in G$   
25       with probability  $1-\eta$  (over the choice of the random oracle  $h$ ). As an example of

unreliable hashing consider hashing onto an elliptic curve  $E : y^2 = g(x)/F_p$ . The set  $A$  can be the field  $F_p \times \{0, 1\}$ , and  $B$  can be the set of points  $x \in A$  for which  $g(x)$  is a quadratic residue in  $F_p$ .

An  $\eta$ -unreliable hash function  $h'$  can be used to construct a reliable hash function  $h$  onto  $G$ . Fix a small parameter  $I = \lceil \log_2 \log_{1-\eta} \delta \rceil$ , where  $\delta$  is a desired bound on the probability of failure.

For any  $i \in \{0, \dots, 2^I - 1\}$ , let  $x_i$  be the output of  $h'(i \parallel m)$ , where  $I$  is represented as an  $I$ -bit string. Find  $i^*$ , the smallest  $i$  for which  $x_i \neq \perp$ . The hash  $h(m)$  of a message  $m$  is defined to be  $x_{i^*}$ .

For each  $i$ , the probability that  $x_i$  is a point on  $G$  is  $\eta$ , so the expectation on calls to  $h'$  is  $1/\eta$ , and the probability that a message  $m$  will be found unhashable is  $(1-\eta)^{2^I} \leq \delta$ . Note, also, that  $h$  is collision-resistant if  $h'$  is, since a collision on  $h$  necessarily exposes a collision on  $h'$ .

Given an unreliable hash function  $h'$ , and an integer  $I$  as parameters, one may define the algorithm *MapToGroup*, which maps arbitrary input strings onto  $G$  with overwhelming probability. An exemplary algorithm works as follows:

- (1) given  $x \in \{0, 1\}^*$ , set  $i \leftarrow 0$ ,
- (2) set  $y \leftarrow h'(i \parallel x)$ ,
- (3) if  $y \neq \perp$ , return  $y$ ,
- (4) otherwise, increment  $i$  and goto step (2),
- (5) if  $i$  reaches  $2^I$ , report failure.

The failure probability may be made arbitrarily small by picking an appropriately large  $I$ , as above.

## Short Digital Signature Schemes Using More General Curves Having A Genus Greater Than One:

In the case of genus one can use elliptic curves via standard complex multiplication methods so that the curves need not be supersingular. In addition, here, it is shown that super-singular curves of genus 2 or 3, for example, may be used to obtain short digital signatures. Although these curves do not give GDH group as described above, they and others like them may still be used to provide beneficial short digital signatures. Here, for example, one important tool that can be used is Weil pairing on the Jacobian of these curves.

Let  $E/F_p^l$  be an algebraic curve of genus  $g = 2$  or  $g = 3$  and let  $J$  be its Jacobian. Let  $P, Q \in J$  be linearly independent points of order  $q$ . Assume  $P \in J/F_{p^l}$  and  $Q \in J/F_{p^{la}}$ . Using the Weil pairing in  $J$  it is easy to decide if a given tuple  $(P, aP, Q, bQ)$  satisfies  $a = b$ . This is referred to herein as the co-Decision Diffie-Hellman problem, and it has an obvious computational variant: given the tuple  $(P, Q, aQ)$ , compute  $aP$ . Thus, one can modify the GDH digital signature scheme to work in such groups. An exemplary modified (co-gap) digital signature scheme is as follows:

### Key Generation:

Pick  $x \in \mathbb{Z}_q^*$ , and compute  $R \leftarrow xQ$ . The public key is  $R$ ; the secret key is  $x$ .

### Signing:

Given a secret key  $x$ , and a message  $m \in \{0, 1\}^*$ , compute  $P_m \leftarrow h(m) \in J/F_{p^l}$ , and  $S_m \leftarrow xP_m$ . The digital signature  $\sigma$  is the  $x$ -coordinate of the  $g$  points in the representation of  $S_m$  as a reduced divisor.

### Verification:

Given a public key  $R$ , a message  $m$ , and a purported digital signature  $\sigma$ , let  $S$  be a point on  $J/F_p$  whose  $x$ -coordinate is in  $\sigma$  and whose  $y$ -coordinate is  $y$  for some  $y \in F_p$  (if no such point exists reject the digital signature as invalid). Set  $u \leftarrow e(P, S)$  and  $v \leftarrow e(R, \phi(h(m)))$ . If  $u = v$  accept the digital signature, otherwise reject it.

The tests in the verification phase ensure that either  $(P, R, h(m), S)$  or  $(P, R, h(m), -S)$  is a valid co-Diffie-Hellman tuple. While the public key,  $R$ , is an element of  $E/F_{p^a}$ , and thus long, a digital signature  $\sigma$  is an element of  $E/F_p$ , and thus relatively short.

In certain instances, the verification algorithm may not be entirely complete. Here, for example, if the digital signature does not contain the  $y$ -coordinates then one will need to recompute them when verifying the digital signature. However, there are two possible values for the  $y$  coordinate. On a curve of genus  $g$  this means that there are  $2^g$  possibilities for  $S$  (in the verification algorithm). So, one would need to test whether any of these  $2^g$  candidates are a valid digital signature.

The security of such schemes follows from the assumption that no adversary  $(t, \epsilon)$  breaks the co-Computational Diffie-Hellman problem. In certain exemplary implementations, super singular curves of genus 2 and 3 have been constructed.

First, a necessary condition for CDH intractability on a subgroup of  $J$  is characterized.

Let  $p$  be a prime,  $l$  a positive exponent, and  $J$  a Jacobian of some curve over  $F_p$  with  $m$  points, where  $m$  is a small multiple of a prime. Then  $J$  has a security multiplier  $\alpha$ , for some integer  $\alpha > 0$ , if the order of  $p^l$  in  $F_m^*$  is  $\alpha$ .

1 In other words:

2 
$$m \mid p^{l\alpha} - 1 \text{ and } m \nmid p^{lk} - 1 \text{ for all } k=1,2,\dots,\alpha-1$$

3 For a large prime  $q$  dividing  $m$ , so that:

4 
$$q^2 \nmid m$$

5 the Jacobian  $J$  has a security multiplier  $\alpha_q$  for  $q$  if the order of  $p^l$  in  $F_q^*$  is  $\alpha_q$ .

6 By necessity,  $\alpha_q$  divides  $\alpha$  for all curves. For a point  $P$  on  $J$ , with order  $q$ ,  
7 the security parameter  $\alpha_q$  bounds from below the size of fields into which  $\langle P \rangle$  can  
8 be mapped. Consider any nontrivial homomorphism from  $\langle P \rangle$  into a subgroup  $A$   
9 of  $F_{p^i}^*$ . Then  $q$  divides  $|A|$ , and  $|A|$  divides  $|F_{p^i}| = p^i - 1$ . Thus  $q \mid p^i - 1$ , so  $i \geq \alpha_q$ .

10 Let  $J$  be the Jacobian of this curve. This curve of genus 2 has security  
11 multiplier  $\alpha = 12$ . The advantage in using the higher genus curves is that the  
12 security multipliers can be higher. Hence, one needs to find values of  $l$  for which  
13 the number of points on  $J/F_{2^l}$  is a small multiple of a prime. Let  $m(l)$  be the  
14 number of points on  $J/F_{2^l}$ . Here, it is known that  $m(l)$  is an integer of length  $2l$   
15 bits. For  $l = 43$  one can show that  $m(l)$  is a small multiple of a prime. Hence, for  $l$   
16  $= 43$  one gets a digital signature of length 86 bits where breaking the scheme  
17 requires the computation of a discrete log on a subgroup of  $J/F_{2^{43}}$  of size  
18 approximately  $2^{86}$ . Furthermore, when using the Weil pairing to reduce the  
19 discrete log problem to a finite field, one obtains a discrete log problem in the  
20 group  $F_{2^{12+43}} = F_{2^{516}}^*$ .

21 Let  $q$  be the largest prime factor of  $m(43)$ . Then  $J/F_{2^{43}}$  contains a point  $P$   
22 of order  $q$ . The open problem now is to prove that  $J/F_{2^{516}}$  contains a point  $Q$  of  
23 order  $q$  which is linearly independent of  $P$ . This is needed for verifying digital  
24 signatures. This is needed for verifying digital signatures and is guaranteed to  
25

1 exist by Tate-Honda theory. It is also noted that in certain implementations, for  
2 example, to get  $\alpha = 30$  one might use Abelian varieties that are not Jacobians of  
3 curves.

4 Thus, short digital signature schemes have been presented based on super  
5 singular hyperelliptic curves, for example. The length of the resulting digital  
6 signature is one element in the Jacobian of the curve. By comparison, standard  
7 digital signatures based on discrete log such as DSA typically require two  
8 elements.

## 9 10 Conclusion

11 Although the description above uses language that is specific to structural  
12 features and/or methodological acts, it is to be understood that the invention  
13 defined in the appended claims is not limited to the specific features or acts  
14 described. Rather, the specific features and acts are disclosed as exemplary forms  
15 of implementing the invention.